ELSEVIER

Artificial Intelligence 91 (1997) 155–171

Artificial Intelligence

k-Certainty Exploration Method: an action selector to identify the environment in reinforcement learning

Kazuteru Miyazaki *, Masayuki Yamamura [1], Shigenobu Kobayashi [2]

Department of Computational Intelligence and Systems Science,
Interdisciplinary Graduate School of Science and Engineering,
Tokyo Institute of Technology, 4259, Nagatsuta, Midori-ku, Yokohama, 226 Japan

Received April 1996; revised July 1996

Abstract

Reinforcement learning aims to adapt an agent to an unknown environment according to rewards. There are two issues to handle delayed reward and uncertainty. Q-learning is a representative reinforcement learning method. It is used in many works since it can learn an optimum policy. However, Q-learning needs numerous trials to converge to an optimum policy. If the target environments can be described in Markov decision processes, we can identify them from statistics of sensor–action pairs. When we build the correct environment model, we can derive an optimum policy with the *Policy Iteration Algorithm*. Therefore, we can construct an optimum policy through identifying environments efficiently.

We separate the learning process into two phases: identifying an environment and determining an optimum policy. We propose the *k-Certainty Exploration Method* for identifying an environment. After that, an optimum policy is determined by the Policy Iteration Algorithm. We call a rule *k-certainty* if and only if it has been selected *k* times or more. The *k*-Certainty Exploration Method excepts any loop of rules that already achieve *k*-certainty. We show its effectiveness by comparing it with Q-learning in two experiments. One is Sutton's maze-like environment, the other is an original environment where an optimum policy varies according to a parameter. © 1997 Elsevier Science B.V.

Keywords: Reinforcement learning; Q-learning; Markov decision processes; Policy Iteration Algorithm; *k*-Certainty Exploration Method

* Corresponding author. E-mail: teru@fe.dis.titech.ac.jp.
[1] E-mail: my@dis.titech.ac.jp.
[2] E-mail: kobayasi@dis.titech.ac.jp.

PII S0004-3702(96)00062-8

1. Introduction

Reinforcement learning (RL) is a kind of machine learning. It aims to adapt an agent to an unknown environment with a clue to rewards. There are two issues to handle: delayed reward and uncertainty. The purpose of RL is to acquire an optimum policy efficiently. Yamamura [23] divides RL systems into two approaches. One is the *exploitation intensive type* and the other is the *exploration intensive type*. The exploitation intensive type emphasizes an experience of getting a reward. Profit sharing [4,5,8,11,12] is a representative exploitation intensive type. Though it can learn a rational policy quickly, it does not always guarantee the optimality.

The exploration intensive type emphasizes collection of all experiences to guarantee the optimality. We regard Q-learning [21] as the exploration intensive type since it can learn an optimum policy in Markov decision processes (MDPs). Though Q-learning has attracted many researchers (for example, Clouse and Utgoff [3], Mahadevan and Connell [10], Takemichi and Kakazu [17], Unemi et al. [20], and Whitehead and Ballard [22]), it needs numerous trials to identify the environment precisely. As the environment becomes more complex, the number of trials increases remarkably.

In MDPs, the environment can be statistically identified through the gathering of samples of actions. If the environment has been identified precisely, a *Policy Iteration Algorithm* (PIA) [1], that is a computational procedure based on dynamic programing, can be used to find an optimum policy [15]. Combining PIA with an algorithm which can identify the environment efficiently, we can achieve the purpose of RL.

If we want to identify the environment efficiently, we should give a priority to an action whose number of selection times is the fewest. The *Interval Estimation Method* [6] is one of such algorithms. It may select useless actions in point of identification of the environment because it gives the same priority for an action which has been contributed to getting a reward. To make matters worse, it cannot handle delayed reward.

In this paper, we propose an algorithm which can identify the environment efficiently. We show its effectiveness by comparing it with Q-learning in two experiments.

Section 2 describes the problem, the method, and notations. Section 3 proposes that the *k-Certainty Exploration Method* is an action selector to identify the environment efficiently. Section 4 shows numerical examples to reveal the effectiveness of the *k*-Certainty Exploration Method.

2. The domain

2.1. The target problem

Consider an agent in some unknown environment. At each time step, it gets information about the environment through its sensors and chooses an action to take. As a result of some sequence of actions, it gets a reward from the environment. A pair of a sensory input and an action is called a *rule*. Executing a rule is called a *trial*. The function that maps sensory inputs to actions is called a *policy*. The policy which maximizes the expected reward per action is called an *optimum policy*.

Table 1
Classification of the existing work

variables	state transitions	
	Markov	non-Markov
discrete	Class 1	Class 3
continuous	Class 2	Class 4

In general, it must learn based on imperfect information because its sensory input is limited to a part of the environment. Furthermore, it must handle the delayed reward because a correct action does not always lead to getting a reward immediately. In addition to these issues, we can classify the existing work in RL as shown in Table 1.

There are a lot of works that assume the property of state transitions is *Markovian*. In this case, the environment is treated as stochastic processes, where a sensory input corresponds to some state and an action to some state transition operator. Bradtke [2] extends RL into the *Linear Quadratic Regulation* (LQR) which belongs to non-Markovian environments. Though the works of Tan [18] and Tenenberg et al. [19] treat some dynamical environment, they are numerical examples.

Many works assume that the range of variables is discrete. In this case, the agent senses a set of discrete attribute-value pairs and performs an action in some discrete varieties. Though the work of Lin [9] treats continuous variables, it remains prototype level. Bradtke [2] investigates continuous variables in the LQR domain. Kimura et al. [7] propose *Stochastic Hill Climbing* which assures the rationality in continuous environments.

Here, we treat Class 1 in Table 1. Class 1 is a good domain to show the effectiveness of our method since there are a lot of works in this class. In the next subsection, we describe the general framework of the RL system. After showing the position of traditional works, we will describe our approach.

2.2. *The general framework and our approach*

In this paper, we divide the RL system into three parts, *State recognizer*, *Action selector* and *Learner* (Fig. 1). After the state recognizer senses the environment, *a conflicting set* of rules is made which match with the current sensory input. The action selector decides the rule to execute to the environment from the conflicting set. The learner reinforces some parameters to get more rewards based on the executed actions and acquired rewards. The design targets of RL systems in Class 1 are restricted to the action selector and the learner since the state recognizer is given a priori.

The purpose of RL is to acquire an optimum policy efficiently. Therefore, we must pursue the optimality and the efficiency. In this paper, we classify RL systems into two approaches.

An approach that pursues the optimality is called the exploration intensive type, where it should be emphasized to identify the environment efficiently. Q-learning is known as a representative RL system. We classify it into the exploration intensive type since it guarantees the optimality in MDPs. We show the framework of Q-learning in Fig. 2.

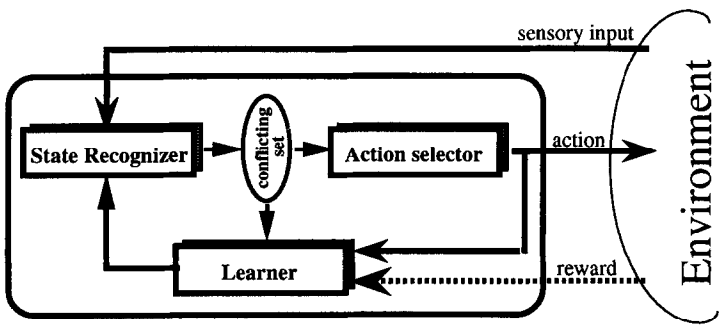

Fig. 1. Framework of reinforcement learning systems.

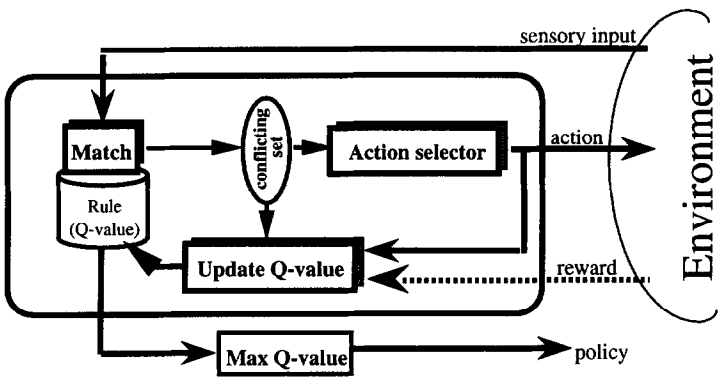

Fig. 2. Framework of learning systems baed on Q-learning.

In Q-learning, a *Q-value* is accumulated to every state–action pair through interactions with the environment. Q-learning needs numerous trials to acquire an optimum policy, because it does not directly identify the environment but it evaluates it through Q-values. To make matters worse, we do not know how to decide action selectors to accelerate the convergence of Q-values.

An approach that pursues the efficiency is called the exploitation intensive type, where it makes many accounts of experiences in getting a reward. Profit sharing is a representative method of exploitation intensive types. Though it can learn a rational policy quickly under some conditions [11, 12], it does not always acquire an optimum policy.

In Class 1, if the environment has been identified precisely, PIA can be used to find an optimum policy. Therefore, if we can build an algorithm which identifies the environment efficiently, the purpose of RL can be achieved easily.

In the next section, we propose the *k-Certainty Exploration Method* which is an action selector to identify the environment efficiently. Combining the *k*-Certainty Exploration Method with PIA, it is possible to acquire an optimal policy efficiently.

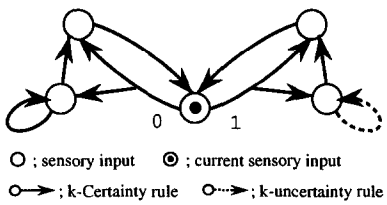

Fig. 3. An example of a k-certain looped rule.

3. k-Certainty Exploration Method

3.1. The basic idea

We propose an algorithm which can identify the environment efficiently. It is important for efficient identification of the environment to select all rules uniformly and as many as possible.

We call a rule *k-certainty* if and only if it has been selected k times or more. We call a rule which is not k-certainty *k-uncertainty*. A sensory input is regarded as a *state*. We call a current sensory input as a *current state* and a sensory input which has been sensed until that time as a *known state*.

Considering a case which returns to a current state after selection of a k-certain rule in the current state. We call a rule a *k-certain looped rule* if and only if the rule leads surely to a k-certain loop which is constructed only by k-certain rules. For example, though rule 0 and rule 1 in Fig. 3 can select in the current state, only rule 1 is a *k-certain looped rule*.

If we select a k-certain looped rule, it is interrupted to uniform selection of rules. It is the basic concept in this paper to except k-certain looped rules from a candidate for selection.

In the next subsection, we propose the k-Certainty Exploration Method which is an action selector to identify the environment efficiently through exception of k-certain looped rules. Furthermore, we propose a learning system based on the k-Certainty Exploration Method to learn an optimum policy.

3.2. The proposed method

The algorithm of the k-Certainty Exploration Method is shown in Fig. 4. It consists of an action selector based on k-Certainty Exploration and updating a certainty level k which controls the identification level of the environment.

k-Certainty Exploration. k-Certainty Exploration builds a maximum likelihood model of the state transition probabilities $\overline{q_{ij}^a}$ and the expectation of rewards $\overline{r_i^a}$. $\overline{q_{ij}^a}$ and $\overline{r_i^a}$ are estimated by

```
procedure k-certainty exploration method

begin
 if there are any 1-uncertainty rules in current state then k:=1.
 else if  all known rules are k-Certainty then k:=k+1.

 begin
  if there are any k-uncertainty rules in current state then
     select the one of these rules at random.
  else rise all flags in all known states.
   for all states except for current state do
    if there are any k-uncertainty rules or
       rules that can transfer to the state whose flag is down then
      be down the flag in the state.
   while there is a state whose flag is down.
   select the one of rules that can transfer to the state
   whose flag is down at random.
  end.

end;
```

Fig. 4. Algorithm of the k-Certainty Exploration Method.

$$\overline{q_{ij}^a} = \frac{\text{the number of transitions from state } i \text{ to } j \text{ by executing rule } a}{\text{the number of executing rule } a \text{ in state } i},$$

$$\overline{r_i^a} = \frac{\text{the sum of immediate rewards by executing rule } a \text{ in state } i}{\text{the number of executing rule } a \text{ in state } i}.$$

k-Certainty Exploration uses flags to except k-certain looped rules from a candidate for selection. A flag is assigned to each state. The number of flags is unknown in advance because the number of state in the environment is unknown. We show the algorithm of k-Certainty Exploration below.

If there are k-uncertain rules in a current state, one of these rules is selected at random. Otherwise, it is necessary to avoid repeatedly selecting k-certain rules. To do so, all flags of known states are raised. Flags of states that can select k-uncertain rules are got down. Furthermore, flags of states that can transfer to flag-down states are also got down. This process is continued until no flag can be got down. Rules that cannot transfer to flag-down states can not meet any k-uncertain rules. Therefore, the algorithm selects at random one of the rules which can transfer to the flag-down states.

How to update k. The k-Certainty Exploration Method controls the identification level of the environment by a certainty level k. Initially, k is set to 1. If all *known rules*, which can select in states how to transit it, become k-certainty, k is set to $k+1$. If a new state is perceived, k is reset to an initial value. Because the sampling number of the new state should be increased selectively.

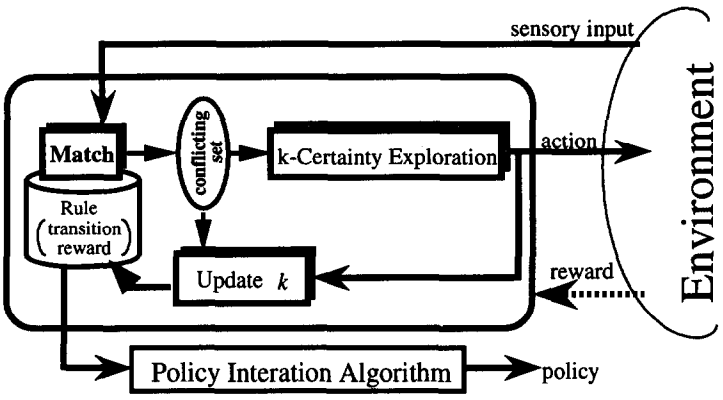

Fig. 5. Framework of learning systems based on k-Certainty Exploration Method.

A learning system based on the k-Certainty Exploration Method. We show a learning system based on the k-Certainty Exploration Method in Fig. 5. It is divided into two parts, identifying the environment and deciding a policy. The part of identifying the environment is the k-Certainty Exploration Method. The part of deciding a policy is PIA.

When all rules become k-certainty, PIA is applied. This policy is reliable in the sense that all rules select at least k times.

3.3. An example

We show a behavioral example of the k-Certainty Exploration Method. Consider the case that the agent can execute two actions and it senses S0 for the first time (Fig. 6—1st).

Initially, k is set to 1. Since all rules are 1-uncertain, for example rule 0 is selected at random. Then, a new sensory input S3 has been sensed (Fig. 6—2nd). Since all rules are 1-uncertain in S3, for example rule 2 is selected at random and S3 has been sensed again (Fig. 6—3rd). Rule 2 is already 1-certain in S3. Therefore, rule 3 is sure to be selected. As a result, S0 has been sensed and it gets a reward at the same time. Similarly, after rule 1 and rule 5 are selected (Fig. 6–4th and 5th), S0 has been sensed again (Fig. 6—6th).

Since all rules in S0 are 1-certain, it cannot decide which rule should be selected in S0. In this case, the k-Certainty Exploration Method works to except 1-certain looped rules from a candidate for selection. First, all flags of known states, *S0*, *S1 and S3*, are raised. The flag of S1 is put down because of the existence of a 1-uncertain rule in S1. Therefore, it selects rule 1 and senses S1 (Fig. 6—7th). Note that there is a case that another sensory input has been sensed because of stochastic state transitions. Since rule 5 is already 1-certain in S1, rule 6 is sure to be selected. As a result, a new

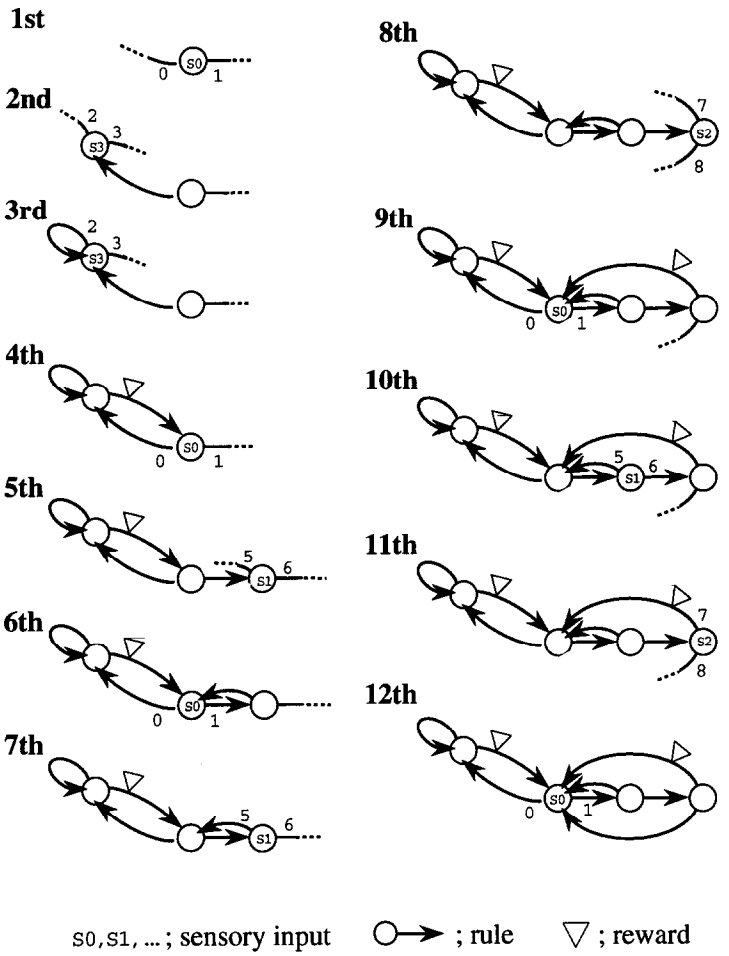

Fig. 6. Execution of the *k*-Certainty Exploration Method.

sensory input S2 has been sensed (Fig. 6—8th). Since all rules are 1-uncertain in S2, for example rule 7 is selected at random. It gets a reward at that time and S0 has been sensed again (Fig. 6—9th).

Since all rules in S0 are 1-certain, the *k*-Certainty Exploration Method works to except 1-certain looped rules. First, flags in S0, S1, S2 and S3 are raised. The flag of S2 is put down because of the existence of a 1-uncertain rule in S2. Furthermore, the flag of S1 is put down because S1 can transfer to S2 which is already in the flag-down state. Therefore, it selects rule 1 and senses S1 (Fig. 6—10th). Similarly, in S1, the *k*-Certainty Exploration Method works to except 1-certain looped rules. Then, rule 6 has been selected and S2 has been sensed again (Fig. 6—11th). Since rule 7 is already 1-certain in S2, rule 8 is sure to be selected. Then, S0 has been sensed again (Fig. 6—12th).

All rules become 1-certain. If we use PIA, we can find an optimum policy which selects rule 0 in S0 and rule 3 in S3. The concrete algorithm of PIA is shown in Appendix A. If we want to update identification levels of the environment, we should repeat the same processes on $k = 2$.

3.4. Features of the k-Certainty Exploration Method

The k-Certainty Exploration Method has the following features:

(1) *Efficient identification of the environment.* We can expect to identify the environment efficiently to except k-certain looped rules from a candidate for selection.

(2) *Complete identification of the environment in deterministic MDPs.* If all rules become 1-certain in deterministic MDPs, the environment has been identified completely.

(3) *Progressive identification of the environment in stochastic MDPs.* In stochastic MDPs, we can control identification levels of the environment by updating a certainty level k.

(4) *Independency on reward.* The k-Certainty Exploration Method is not influenced by positions of rewards. It reflects on our belief that using rewards does not always lead to efficient identification of the environment.

(5) *Polynomial computational costs.* The numerical cost of memory is $O(mn^2)$ and the cost of time is $O(mn^3)$ where m is the number of actions and n is the number of sensory inputs. They are polynomial in time. PIA can find an optimum policy in polynomial time too [14].

The k-Certainty Exploration Method has many features. In the next section, we show the effectiveness of the k-Certainty Exploration Method by comparing it with other methods.

4. Evaluation of k-Certainty Exploration Method

4.1. Estimation in a multi-return environment

The k-Certainty Exploration Method identifies the environment efficiently to except k-certain looped rules from a candidate for selection. We confirm the effectiveness of this feature.

By the way, we can consider two criteria for a general learning system. One is to acquire an optimum policy efficiently, the other is to decrease numerical costs for deciding the next action. The former corresponds to decreasing the number of trials, the latter to decreasing the time spent to decide the next action. In general, for RL systems, the former is important. Because the trial in an unknown environment is accompanies by risks which cannot be predicted easily. On the other hand, numerical costs are easy to predict. Therefore, we evaluate RL systems based on the number of trials.

We call an action selector which always selects the fewest sampling rule in the current state a *Minimal Select Method.* Though the k-Certainty Exploration Method is a slightly more complex method, the Minimal Select Method is a very simple method. However, it shows an exponential explosion on the number of trials in some environment.

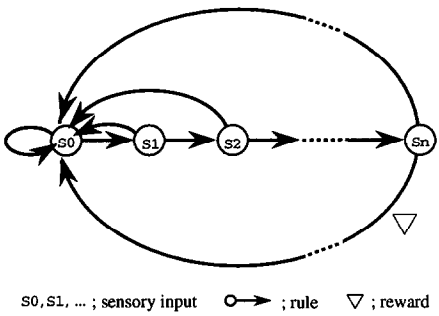

Fig. 7. A multi-return environment.

Considering the environment shown in Fig. 7, if the agent selects special rules in this environment, it forces a return to S0 immediately. We call this environment the *multi-return environment*. If we want to raise a certainty level k by one, then

$$\begin{aligned} &\tfrac{1}{2}(n+1)(n+2)+n+1 && \text{for the } k\text{-Certainty Exploration Method,} \\ &3\cdot 2^n+\tfrac{1}{2}n-1 && \text{for the Minimal Select Method} \end{aligned}$$

is required as the number of trials.

The multi-return environment can be shown very easily. Therefore, we cannot bear the exponential explosion in the Minimal Select Method.

4.2. Comparison with Q-learning in a maze-like environment

The k-Certainty Exploration Method can identify the environment completely in deterministic MDPs. In this subsection, we confirm the effectiveness of this feature through comparison with Q-learning in the maze-like environment of [16].

4.2.1. A maze-like environment

We show the maze-like environment of [16] in Fig. 8. The agent can distinguish all states. At each time step, it senses one state and selects an action out of *move up*, *move down*, *move right* and *move left*. It cannot move to black walls. When it reaches the goal state G, it can get a reward and returns to the start state S.

The shortest path from S to G needs fourteen steps. There are six variations of the shortest path in this environment. The agent must learn an optimum policy for finding one of them. We compare the number of trials to acquire an optimum policy of a learning system based on the k-Certainty Exploration Method and Q-learning.

4.2.2. Performance of the k-Certainty Exploration Method

We show an average number of trails and the standard deviation to acquire an optimum policy in Table 2. We have made 100 trials with different random seeds.

We can get an optimum policy when all rules have achieved k-certainty. Since there are not any stochastic state transitions in this environment, Table 2 coincides with the

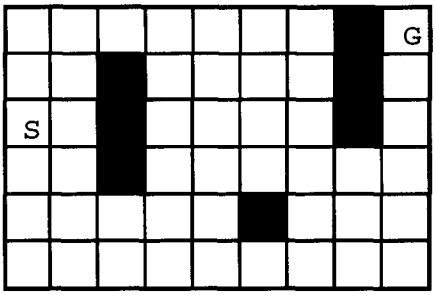

Fig. 8. A maze-like environment.

Table 2
Comparison of the k-certainty Exploration Method and Q-learning on the environment of Fig. 8

	average	standard deviation
k-Certainty Exploration	598.62	397.50
Q-learning	4780.42	2124.46

number of trials to make all rules 1-certainty. The number of trials to make all rules 2-certainty is 1503.5 and the standard deviation is 397.7.

If we use the k-Certainty Exploration Method, the agent moves to black walls at least once in each state. Because the k-Certainty Exploration Method excepts k-certain looped rules. Therefore, we can suppress the explosion of trials to acquire an optimal policy in large state spaces.

4.2.3. Comparison with Q-learning

Q-learning uses the roulette selection in proportion to Q-values for an action selector. Initial Q-values are set to 10.0, the discount factor is 0.9, the reward value is 100.0 and the learning rate of Q-learning is 0.5. These are determined by preliminary experiments.

Q-learning needs correct propagation of rewards to converge Q-values. It needs numerous trials to propagate a reward from G to S since there are many states in this environment. As a result, Q-learning requires about six times as many trials for learning in comparison with a learning system based on the k-Certainty Exploration Method.

Through this numerical example, we can confirm the effectiveness of a learning system based on the k-Certainty Exploration Method in deterministic MDPs.

4.3. Comparison with Q-learning in a stochastic MDP

In stochastic MDPs, the k-Certainty Exploration Method can control the identification level of the environment by updating a certainty level k. Furthermore, it is not influenced by positions of rewards to identify the environment. We confirm these features using an original environment where an optimum policy varies according to a parameter.

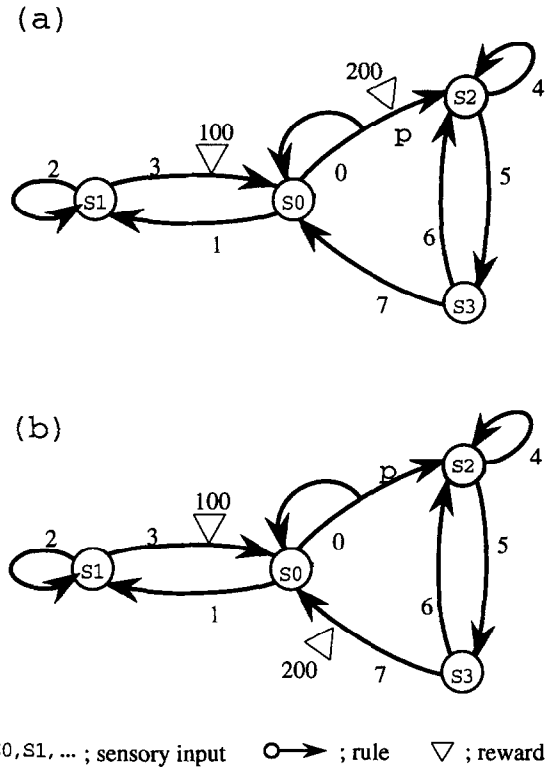

Fig. 9. A stochastic state transition environment.

4.3.1. A stochastic state transition environment

We consider an environment shown in Fig. 9. A parameter p is a state transition probability from S0 to S2 by executing rule 0. In Fig. 9(a), the agent can get 100.0 rewards by executing rule 3 and 200.0 rewards if it selects rule 0 and moves to S2. In Fig. 9(b), it can get 100.0 rewards by executing rule 3 and 200.0 rewards by executing rule 7. If $p < 0.5$, an optimum policy contains rule 1 and rule 3. On the other hand, if $p > 0.5$, it contains rule 0, rule 5 and rule 7. We compare the number of trials to acquire an optimum policy of a learning system based on the k-Certainty Exploration Method and Q-learning under p is 0.1, 0.3, 0.7 and 0.9.

4.3.2. Performance of the k-Certainty Exploration Method

Fig. 10 shows the acquisition rate of the optimal policy plotted against the number of trials in a learning system based on the k-Certainty Exploration Method. We have made 100 trials with different random seeds. Fig. 11 shows the distribution of a certainty level k until the acquisition rate of the optimal policy achieves 1.0.

The acquisition rates of the optimal policy are the same in Figs. 9(a) and (b) because the k-Certainty Exploration Method is not influenced by positions of rewards. Only the state transition probability effects the acquisition rates of it. If p approaches 0.5, the

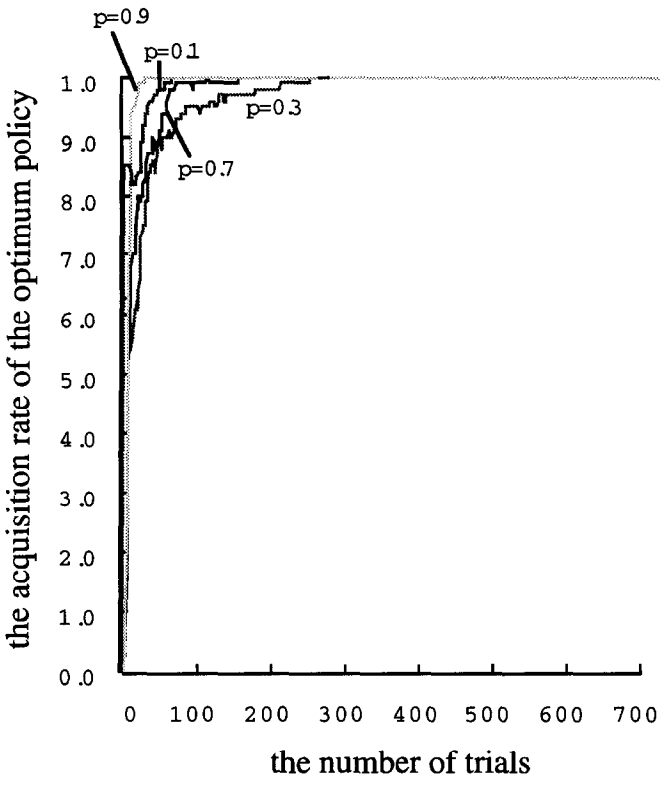

Fig. 10. Behavior of the k-Certainty Exploration Method on the environment of Fig. 9.

agent tends to make a mistake since 0.5 is a boundary of the optimal policy. In this case, the k-Certainty Exploration Method raises a certainty level k. Therefore, distribution of a certainty level k in $p = 0.3$ and $p = 0.7$ spreads out in comparison with $p = 0.1$ and $p = 0.9$ as shown in Fig. 11.

4.3.3. Comparison with Q-learning

Figs. 12 and 13 show the acquisition rate of the optimal policy plotted against the number of trials in Q-learning under the environment of Figs. 9(a) and (b), respectively. Q-learning uses the roulette selection in proportion to Q-values for an action selector. Initial Q-values are set to 10.0, the discount factor is 0.9, the reward value is 100.0 and the learning rate of Q-learning is 0.5. They are determined by preliminary experiments.

In general, Q-learning acquires an optimum policy through Q-values. Q-values reflect the expected reward per action. Therefore, Q-learning is influenced by positions of rewards. In Fig. 9(a), the Q-value of rule 0 is more reinforced than that of rule 1 because of getting a reward by executing rule 0. Therefore, the agent can get the optimum policy easily within $p > 0.5$. On the other hand, in Fig. 9(b), it can get the optimum policy easily within $p < 0.5$ since the Q-value of rule 1 is more reinforced than that of rule 0.

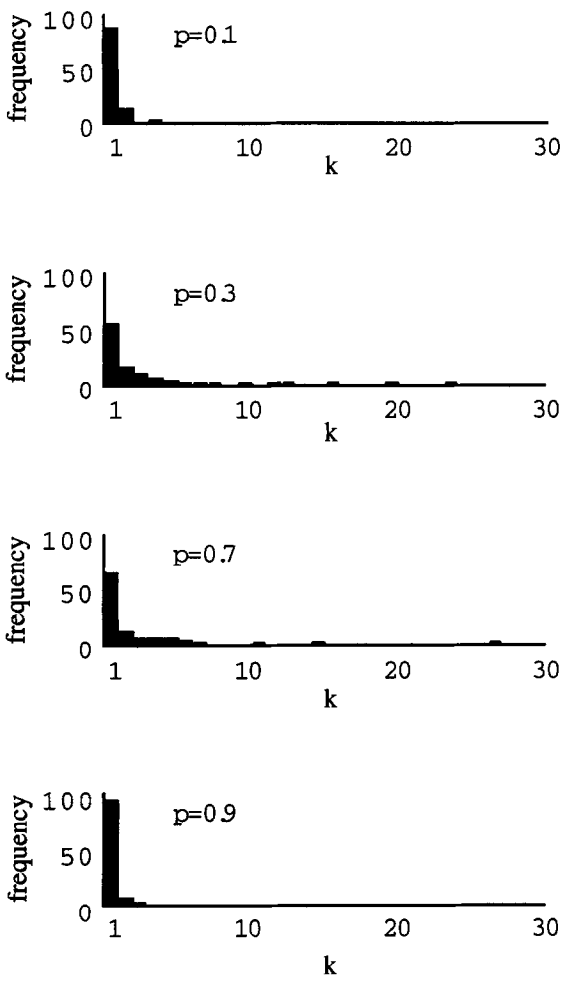

Fig. 11. Frequency distributions of a certainty level k on the environment of Fig. 9.

Q-learning is strictly influenced by positions of rewards. The k-Certainty Exploration Method is not influenced by them. Therefore, it is feasible to some changes of them.

5. Conclusions

We have proposed the k-Certainty Exploration Method which is an action selector to identify the environment efficiently. Combining the k-Certainty Exploration Method with PIA, it is possible to acquire an optimal policy efficiently. We show the effectiveness of the k-Certainty Exploration Method by comparing it with other methods.

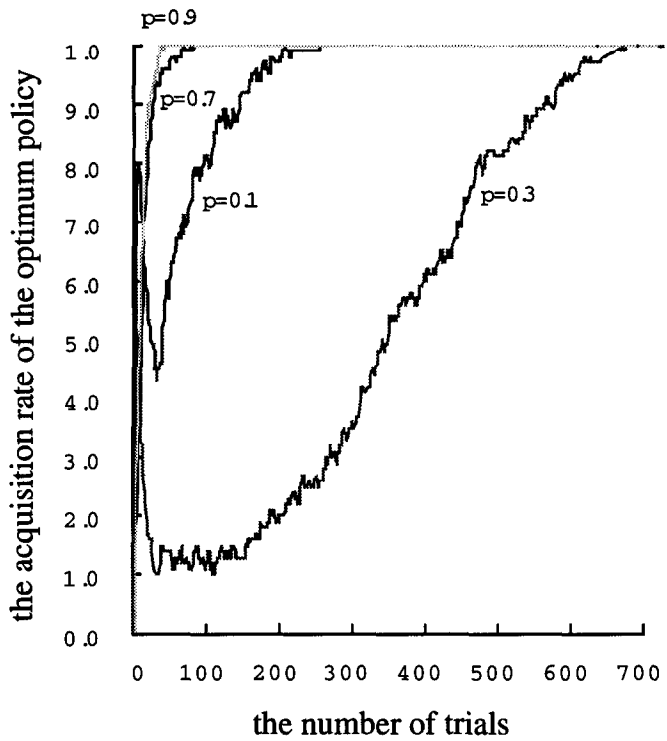

Fig. 12. Behavior of Q-learning on the environment of Fig. 9(a).

The k-Certainty Exploration Method does not consider any state transition probability. Therefore, it does not always guarantee the efficiency under stochastic MDPs. We propose the *ℓ-Certainty Exploration Method* which is an extension of the k-Certainty Exploration Method to stochastic MDPs [13]. The ℓ-Certainty Exploration Method realizes efficient identification of the environment by considering state transition probabilities.

As future work, we want to extend the k-Certainty Exploration Method to non-MDPs. After that, we would like to consider multi-agent reinforcement learning, dynamical environments, multi-reward environments, and so on.

Appendix A

In the following we show the Policy Iteration Algorithm.

Step 1. Select any policy.

Step 2. For that policy, resolve the following equations in w_i, $i = 1, 2, \ldots, m$:

$$w_i = r_i^{k_i} + \gamma \sum_{j=1}^{m} q_{ij}^{k_i} w_j, \quad i = 1, 2, \ldots, m,$$

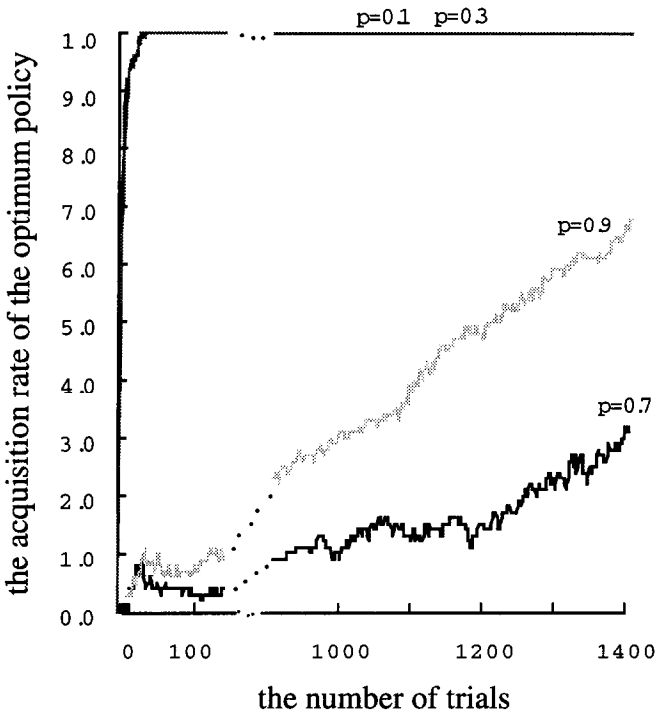

Fig. 13. Behavior of Q-learning on the environment of Fig. 9(b).

where m is the number of states, k_i is an action in state i, γ is a discount factor.

Step 3. Calculate the following equations:

$$W_i = \max_{1 \leqslant k \leqslant n} \left\{ r_i^k + \gamma \sum_{j=1}^{m} q_{ij}^k w_j \right\}, \quad i = 1, 2, \ldots, m,$$

where n is the number of actions.

Step 4. If $W_i = w_i$ for any i, the policy is an optimum policy. If there is an i for which $W_i > w_i$, set the k for which Step 3 is maximized to k_i for any i and return to Step 2.

References

[1] D.P. Bertsekas, Dynamic programming and stochastic control, in: R. Bellman, ed., *Mathematics in Science and Engineering* **125** (Academic Press, New York, 1976).

[2] S.J. Bradtke, Reinforcement learning applied to linear quadratic regulation, in: *Proceedings NIPS-5*, 1993.

[3] J.A. Clouse and P.E. Utgoff, A teaching method for reinforcement learning, in: *Proceedings Ninth International Conference on Machine Learning*, Aberdeen, Scotland (1992) 92–101.

[4] J.J. Grefenstette, Credit assignment in rule discovery systems based on genetic algorithms, *Mach. Learn.* **3** (1988) 225–245.
[5] J.H. Holland and J.S. Reightman, Cognitive systems based on adaptive algorithms, in: D.A. Waterman and F. Hayes-Roth, eds., *Pattern-Directed Inference Systems* (Academic Press, New York, 1987).
[6] L.P. Kaelbling, An adaptable mobile robot, in: *Proceedings 1st European Conference on Artificial Life* (1991) 41–47.
[7] H. Kimura, M. Yamamura and S. Kobayashi, Reinforcement learning by Stochastic Hill Climbing on discounted reward, in: *Proceedings 12th International Conference on Machine Learning*, Lake Tahoe, CA (1995) 295-303.
[8] G.E. Liepins, M.R. Hilliard, M. Palmer and G. Rangarajan, Alternatives for classifier system credit assignment, in: *Proceedings IJCAI-89*, Detroit, MI (1989) 756–761.
[9] L. Lin, Scaling up reinforcement learning for robot control, in: *Proceedings 10th International Conference on Machine Learning*, Amherst, MA (1993) 182–189.
[10] S. Mahadevan and J. Connell, Automatic programming of behavior-based robots using reinforcement learning, in: *Proceedings AAAI-91*, Anaheim, CA (1991) 774–780.
[11] K. Miyazaki, M. Yamamura and S. Kobayashi, A theory of profit sharing in reinforcement learning, *J. Japan. Soc. Artificial Intelligence* **9** 4 (1994) 580–587 (in Japanese).
[12] K. Miyazaki, M. Yamamura and S. Kobayashi, On the rationality of profit sharing in reinforcement learning, in: *Proceedings 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing* (1994) 285–288.
[13] K. Miyazaki, M. Yamamura and S. Kobayashi, ℓ-Certainty Exploration Method: an action selector to identify the environment by an agent—an extension of k-Certainty Exploration Method to stochastic MDPs, *J. Japan. Soc. Artificial Intelligence*, to appear (in Japanese).
[14] C.H. Papadimitriou and J.N. Tsitsiklis, The complexity of Markov decision processes, *Math. Oper. Res.* (1987) 441–450.
[15] S.P. Singh, Transfer of learning by composing solutions of elemental sequential tasks, *Mach. Learn.* **8** (1992) 323–339.
[16] R.S. Sutton, Reinforcement learning architectures for animats, in: *Proceedings 1st International Conference on Simulation of Adaptive Behavior* (1990) 337–343.
[17] Y. Takemichi and Y. Kakazu, An acquisition of action strategy by multi-layered Q-learning, in: *Proceedings 7th Annual Conference of JSAI* (1993) 93–96.
[18] M. Tan, Multi-agent reinforcement learning: independent vs. cooperative agents, in: *Proceedings 10th International Conference on Machine Learning*, Amherst, MA (1993) 330–337.
[19] J. Tenenberg, J. Karlsson and S. Whitehead, Learning via task decomposition, in: *Proceedings 2nd International Conference on Simulation of Adaptive Behavior* (1993) 337–343.
[20] T. Unemi, M. Nagayoshi, N. Hiyayama, T. Nade, K. Yano and Y. Masujima, Evolutionary differentiation of learning abilities—a case study on optimizing parameter values in Q-learning by a genetic algorithm, in: *Proceedings 4th International Workshop on Artificial Life* (1994) 331–336.
[21] C.J.C.H. Watkins and P. Dayan, Technical note: Q-learning, *Mach. Learn.* **8** (1992) 55–68.
[22] S.D. Whitehead and D.H. Ballard, Active perception and reinforcement learning, in: *Proceedings 7th International Conference on Machine Learning*, Austin, TX (1990) 179–188.
[23] M. Yamamura, Reinforcement learning, *J. Japan. Soc. Artificial Intelligence* **8** 6 (1993) 833–834 (in Japanese).